Improving program performance with AMD CodeAnalyst for Linux®

Paul J. Drongowski
AMD CodeAnalyst team

May 9, 2007

AMD CodeAnalyst for Linux®

AMD Smarter Choice

- Identify and investigate performance hot-spots
- Easy to use graphical user interface
  - Simplified configuration of performance events
  - GUI is consistent for cross-platform use
- Low overhead, system-wide data collection
  - Can profile dynamically loaded modules
  - Can profile kernel mode drivers as well as applications
- Results are displayed in configurable views
  - Source and disassembly annotated with performance data
  - Rates and ratios assist interpretation of results

AMD CodeAnalyst for Linux

AMD
Smarter Choice

- Open source
  - Uses oprofile for data collection
  - Can import oprofile data
  - CodeAnalyst GUI is Qt-based
- Java profiling support
  - Version 2.5: Modified oprofile driver and daemon
  - Version 2.6: Modified daemon only

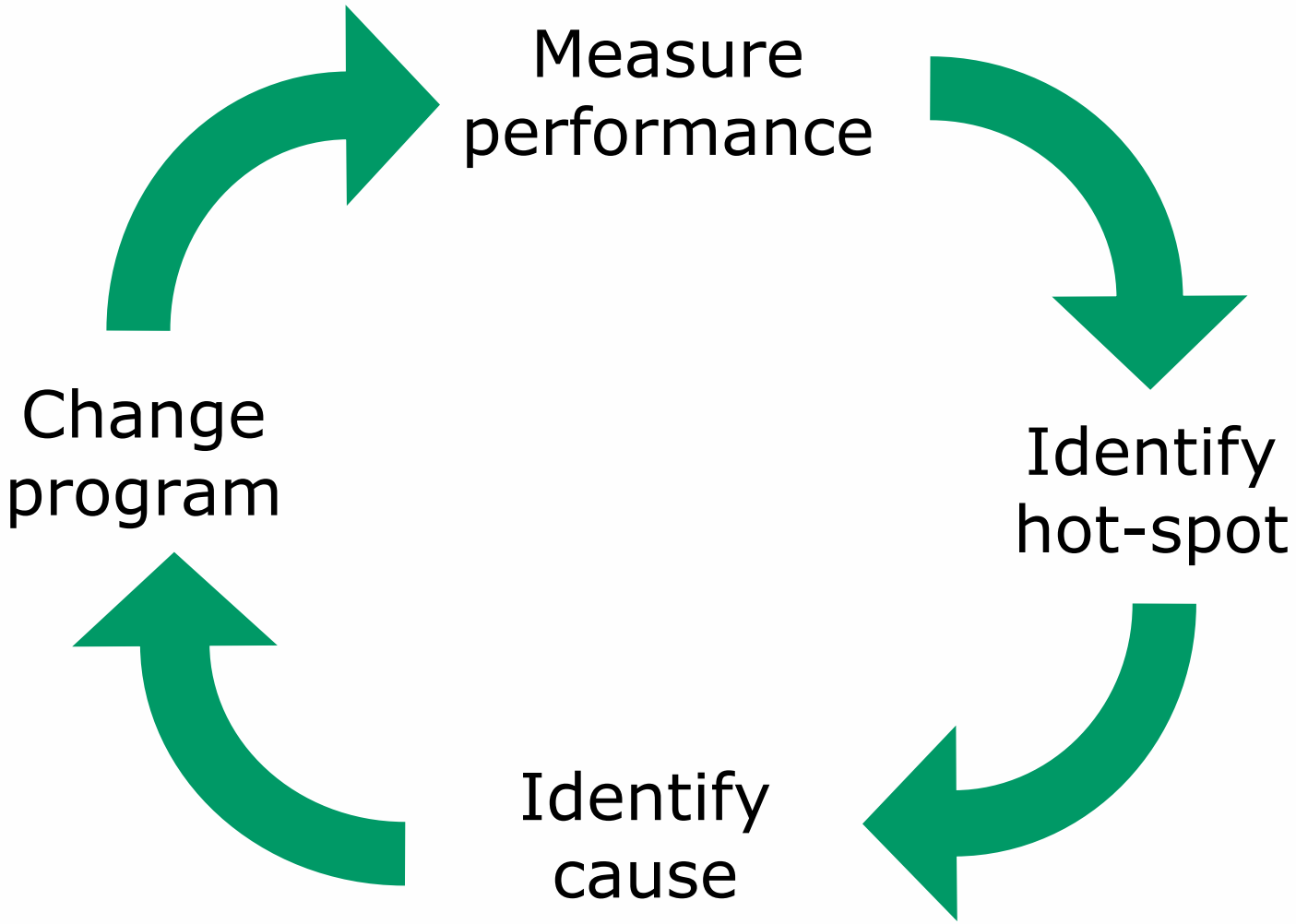
Presentation outline

- AMD CodeAnalyst in action
  - Time-based profiling
  - Event-based profiling
  - Tuning
- Downloading AMD CodeAnalyst
- On-line resources

AMD
Smarter Choice

AMD CodeAnalyst in action

Performance tuning cycle



Kinds of analysis

- Identify hot-spot: Time-based profiling
  - Where is the application or system spending time?
- Identify cause: Event-based profiling
  - How is the application behaving on the CPU? Memory?
  - Are there performance issues in a code region?

Example program

- Classic example: Textbook matrix multiply
- Long memory strides cause data cache and TLB misses

```
void multiply_matrices()
{
  // Multiply the two matrices
  for (int i = 0 ; i < N ; i++) {
    for (int j = 0 ; j < M ; j++) {
      float sum = 0.0 ;
      for (int k = 0 ; k < K ; k++) {
        sum = sum + matrix_a[i][k] * matrix_b[k][j] ;
      }
      matrix_r[i][j] = sum ;
    }
  }
}
```

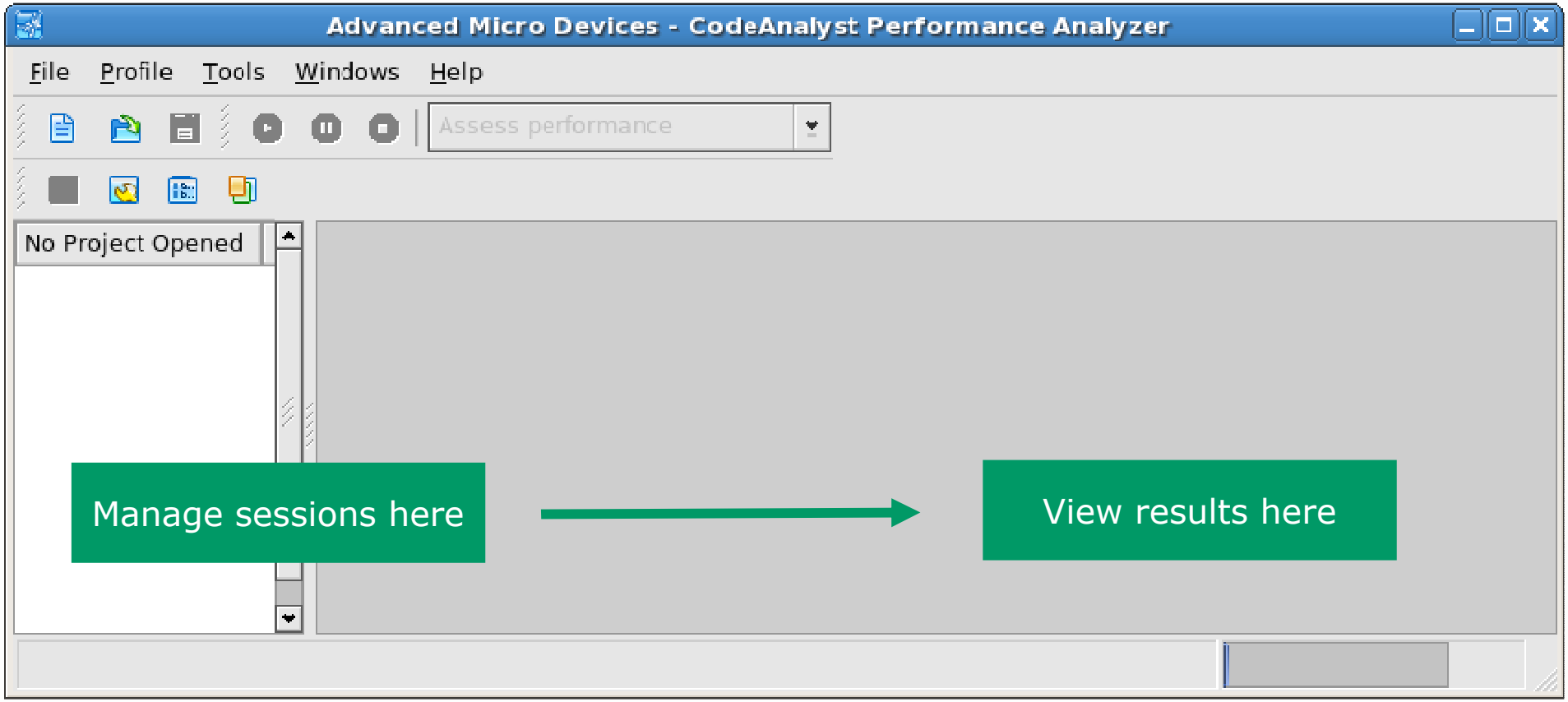
Program preparation

- AMD CodeAnalyst samples all running software components
  - Applications, libraries, kernel modules, ...
- No explicit preparation is necessary
- However, symbolic information is needed to breakdown samples by function or source line
- Compile and build application with debug information

```
g++ -O2 -g -o classic classic.cpp
```

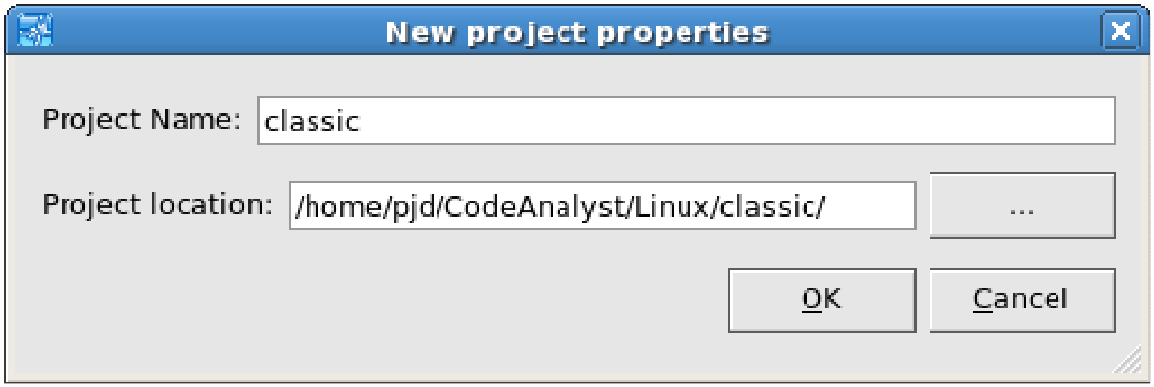
Graphical user interface

- Uses a project- and session-oriented user model
- First step: Create a new project



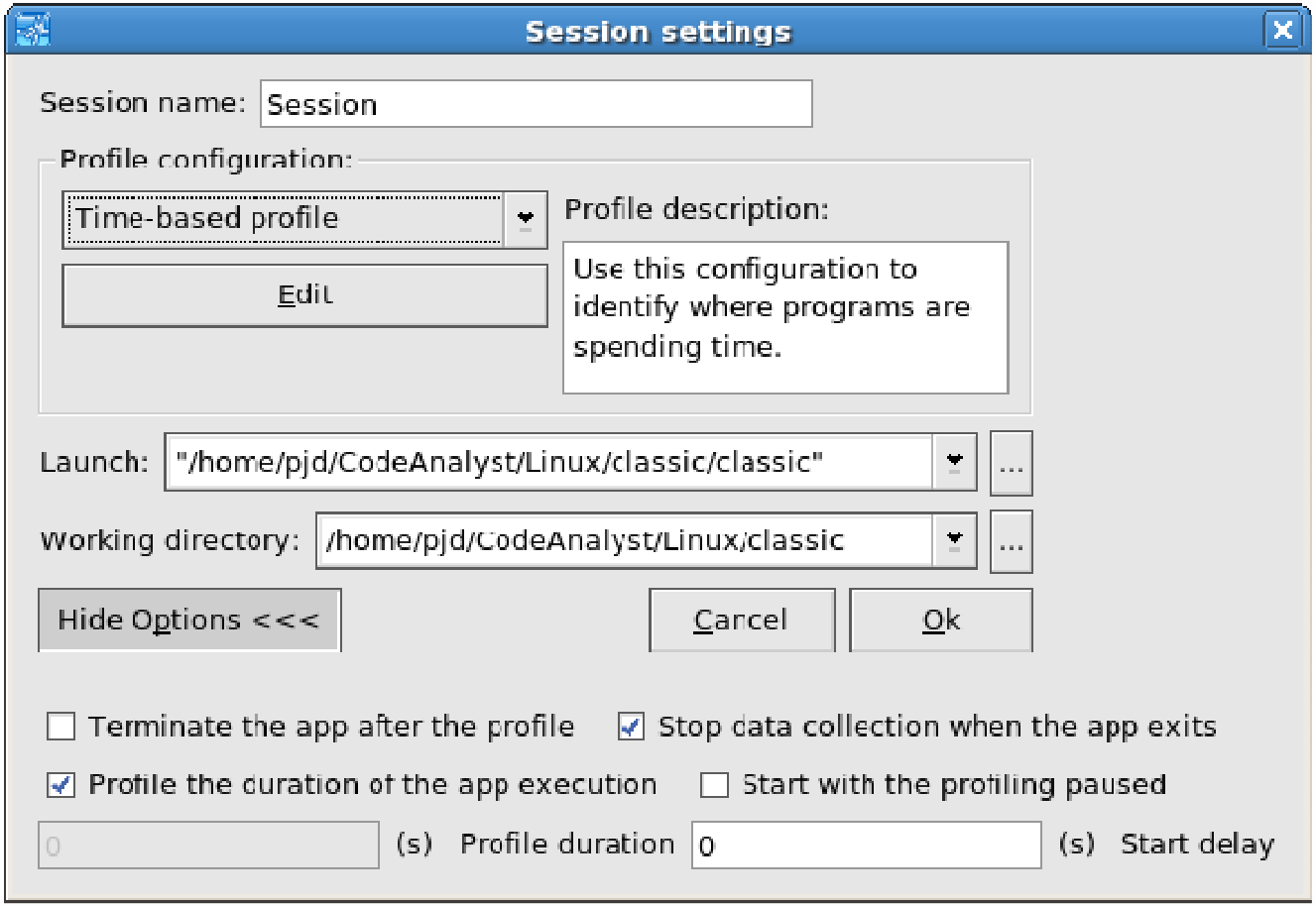
Enter project information

- Choose a name for the new project
- Choose a directory to hold the project



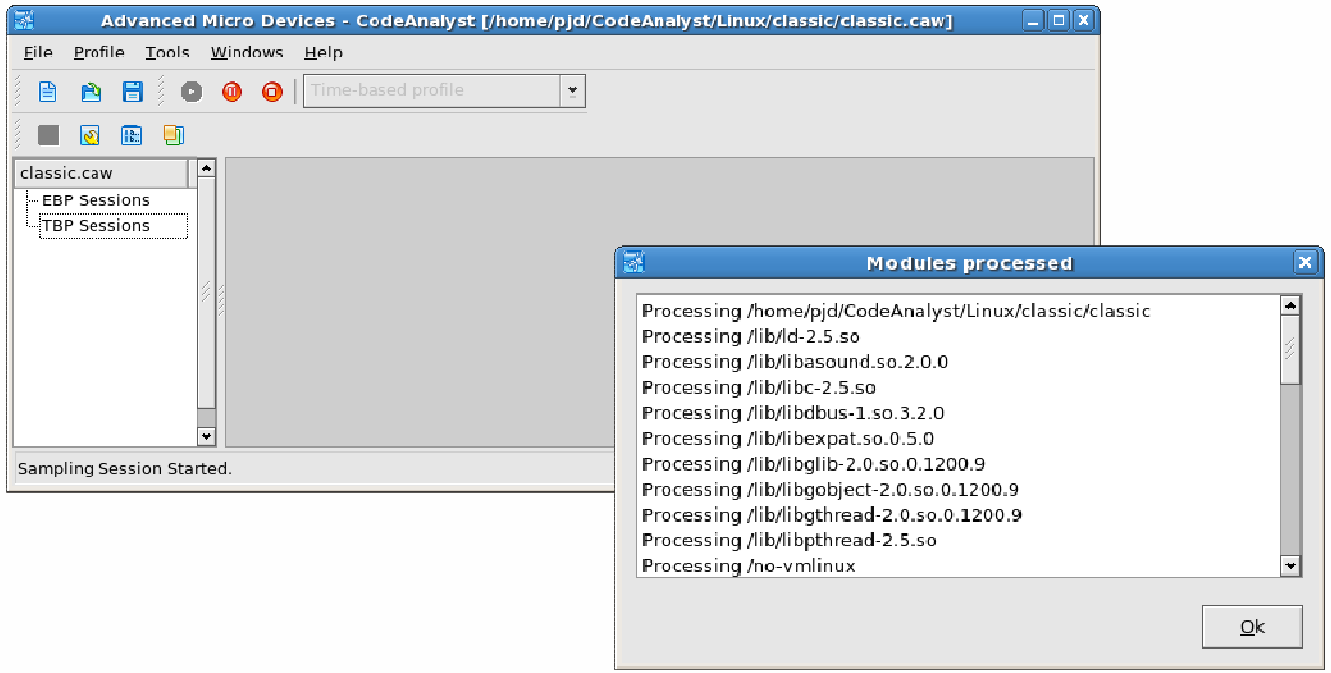
Prepare to collect data

- Enter the path to the application program to analyze and set the working directory
- Choose the kind of analysis to perform



Start data collection

- Click the start button in the toolbar
- AMD CodeAnalyst runs the application and data is collected
- AMD CodeAnalyst imports the data and displays results

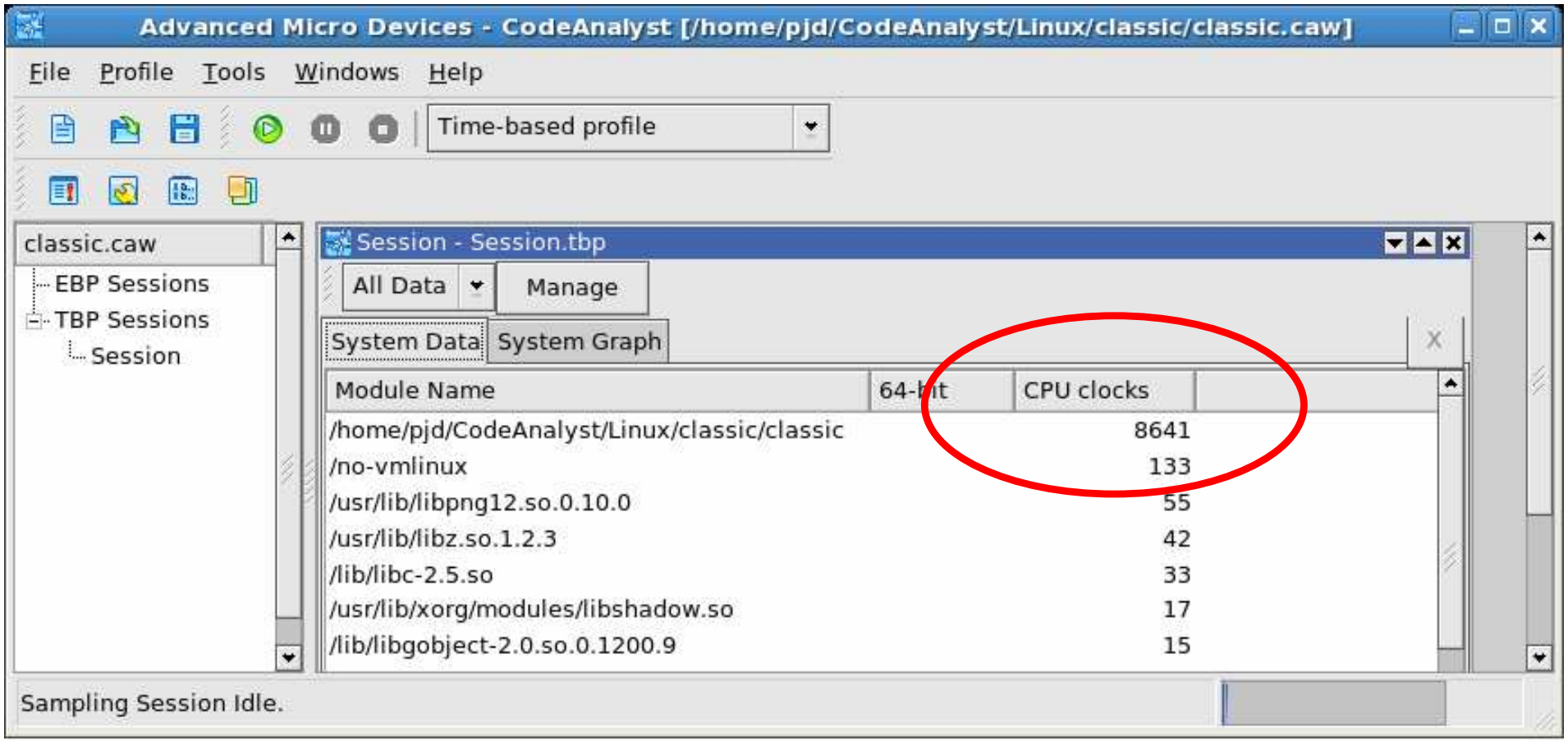


AMD
Smarter Choice

Time-based profiling

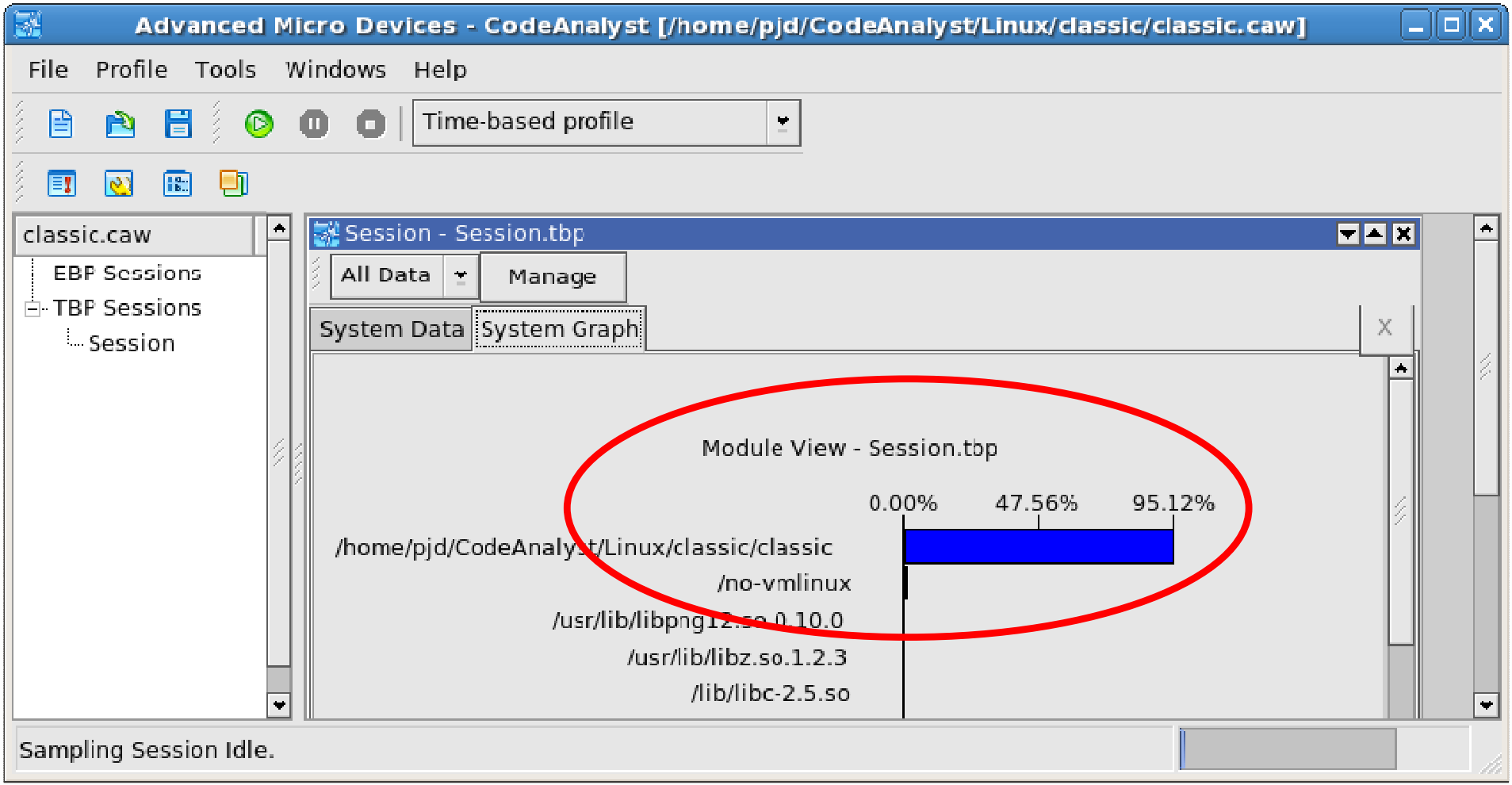
Time-based profiling

- Shows time for each module
- Time is measured by sampling PMC event 0x76 (CPU clocks)
- Double click module to drill down to its functions



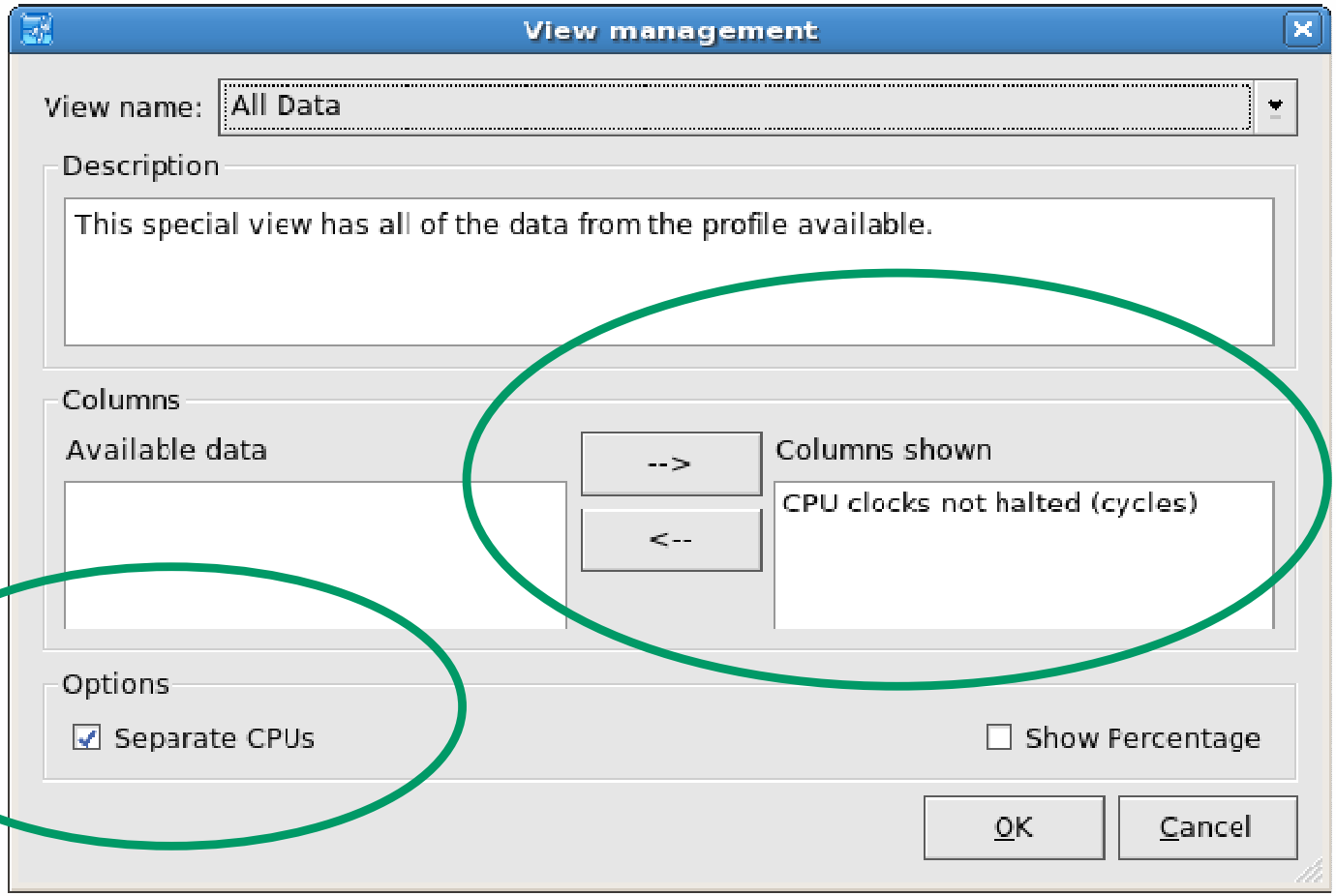
Time-based profiling

- Show results in bar chart form, too



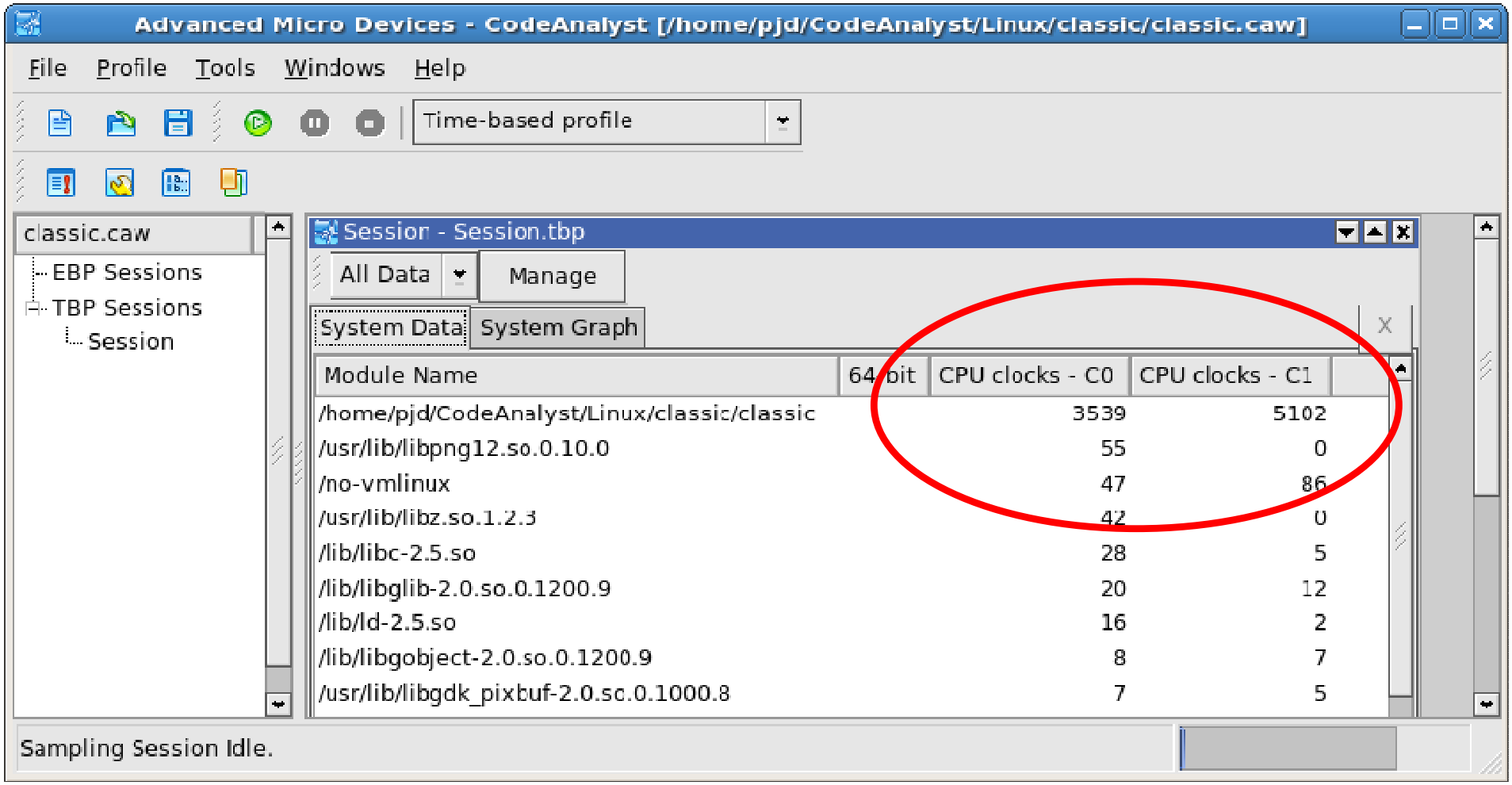
Use "Manage" button to adjust view

- Add or remove columns to be shown
- Aggregate data or separate data by CPU



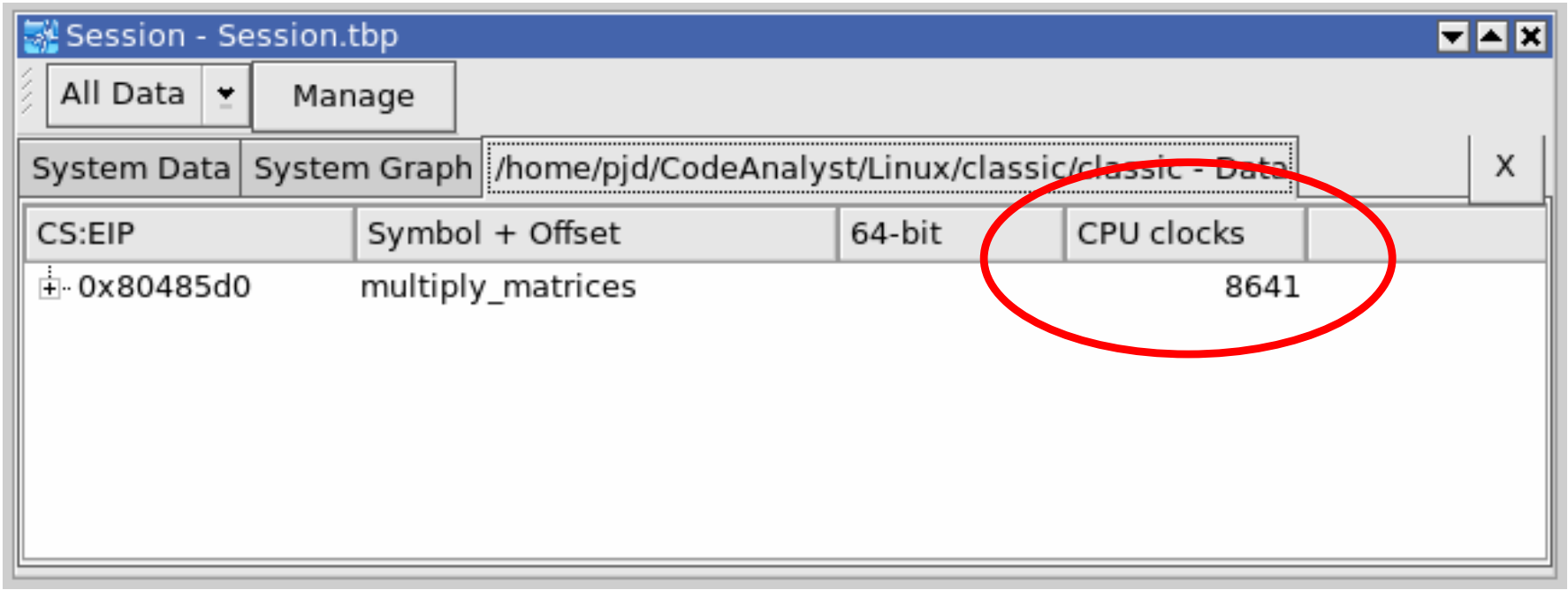
TBP data shown for each CPU

- Shows time spent on each CPU
- Indicates degree of processor affinity during execution



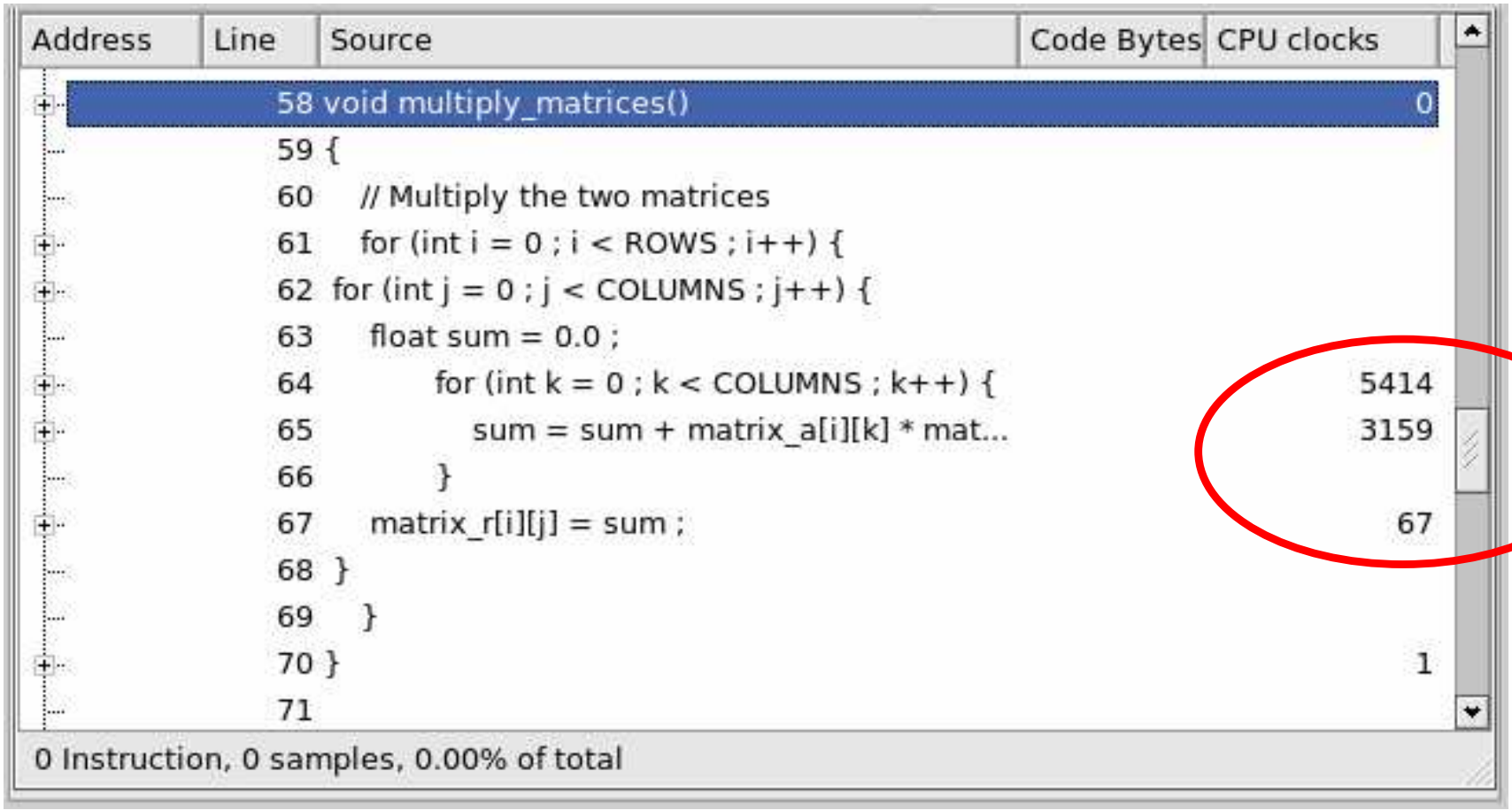
Time-based profiling

- Drill down to functions within a module
- Shows a function-by-function time breakdown
- Identify candidates for investigation and improvement



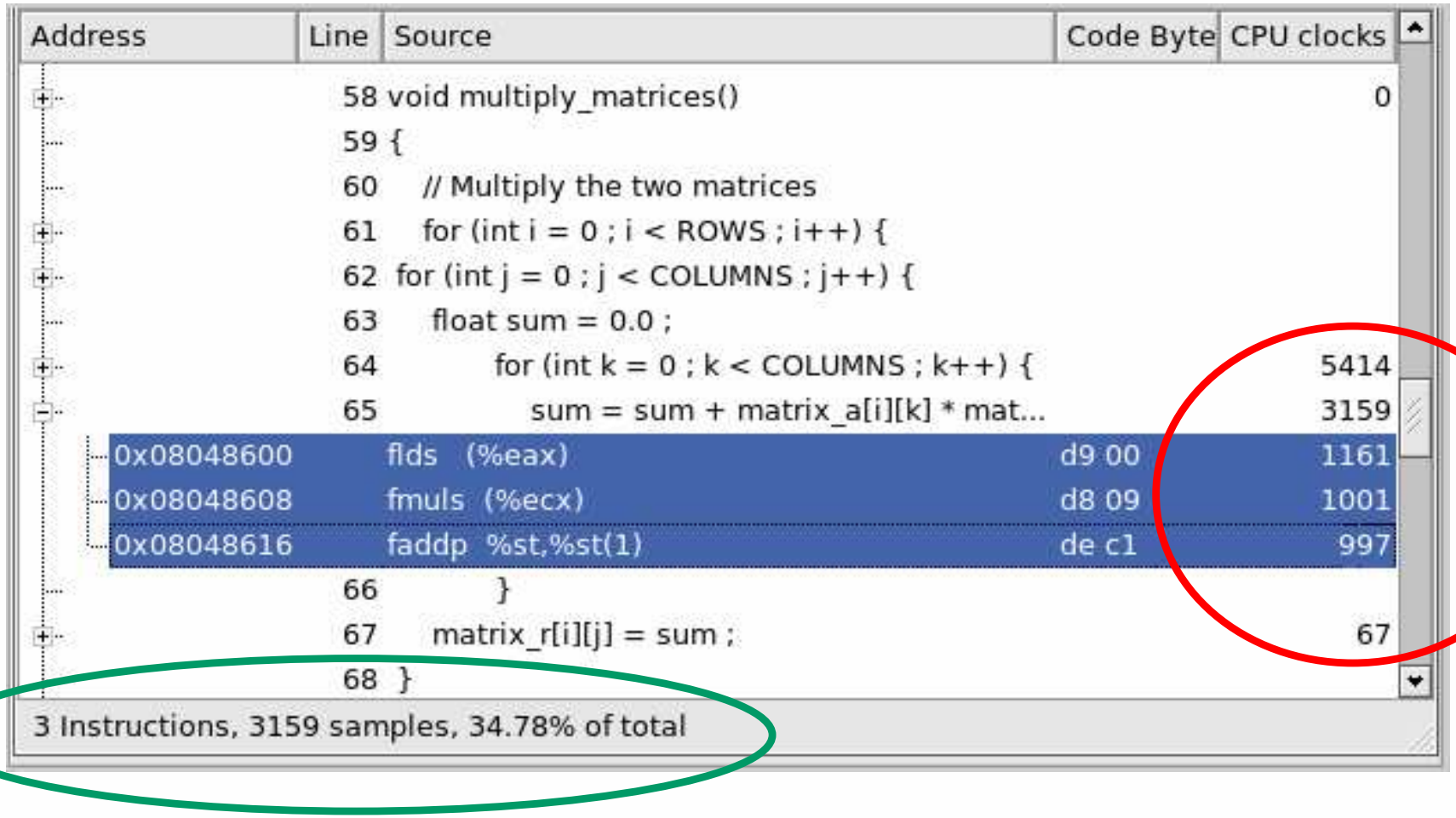
Time-based profiling

- Drill down to source code
- Identify code region to investigate and tune



Time-based profiling

- Drill down to instructions too
- Select instructions to see a summary in the info bar

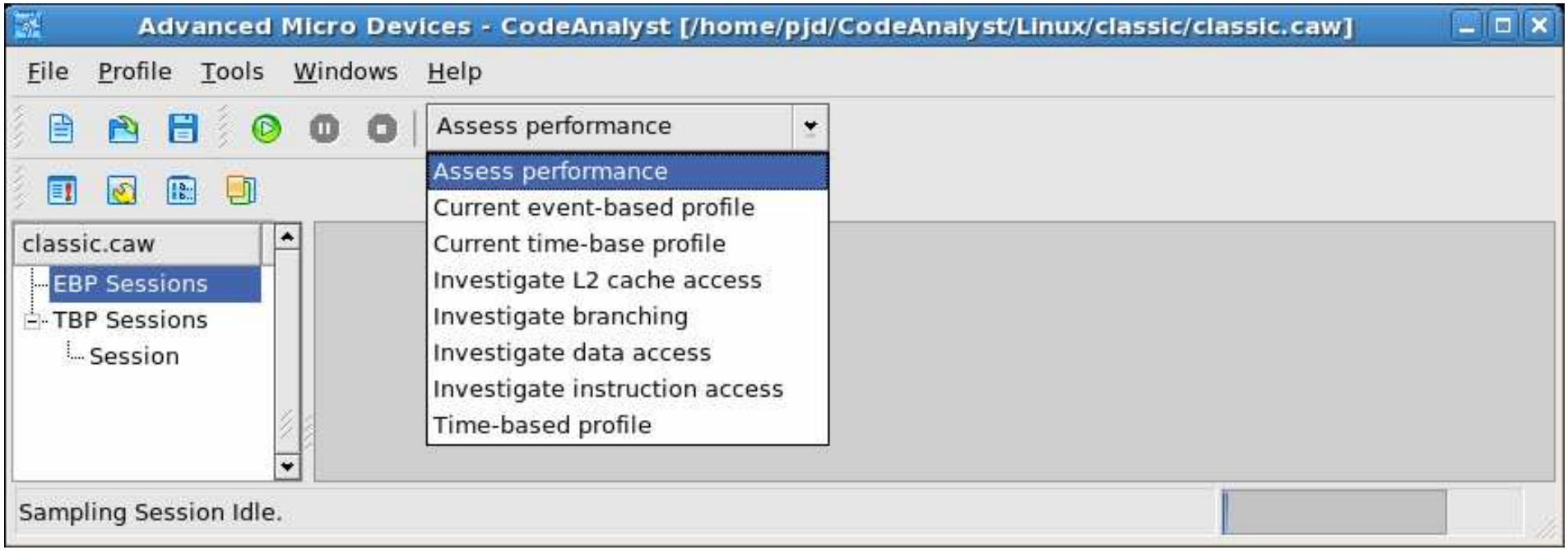


AMD
Smarter Choice

Event-based profiling

Event-based profiling

- Observe how program behaves on CPU and memory
- Develop and test hypothesis about a performance issue
- Choose the type of analysis to perform
- Data display and drill down are similar to TBP



Kinds of analysis

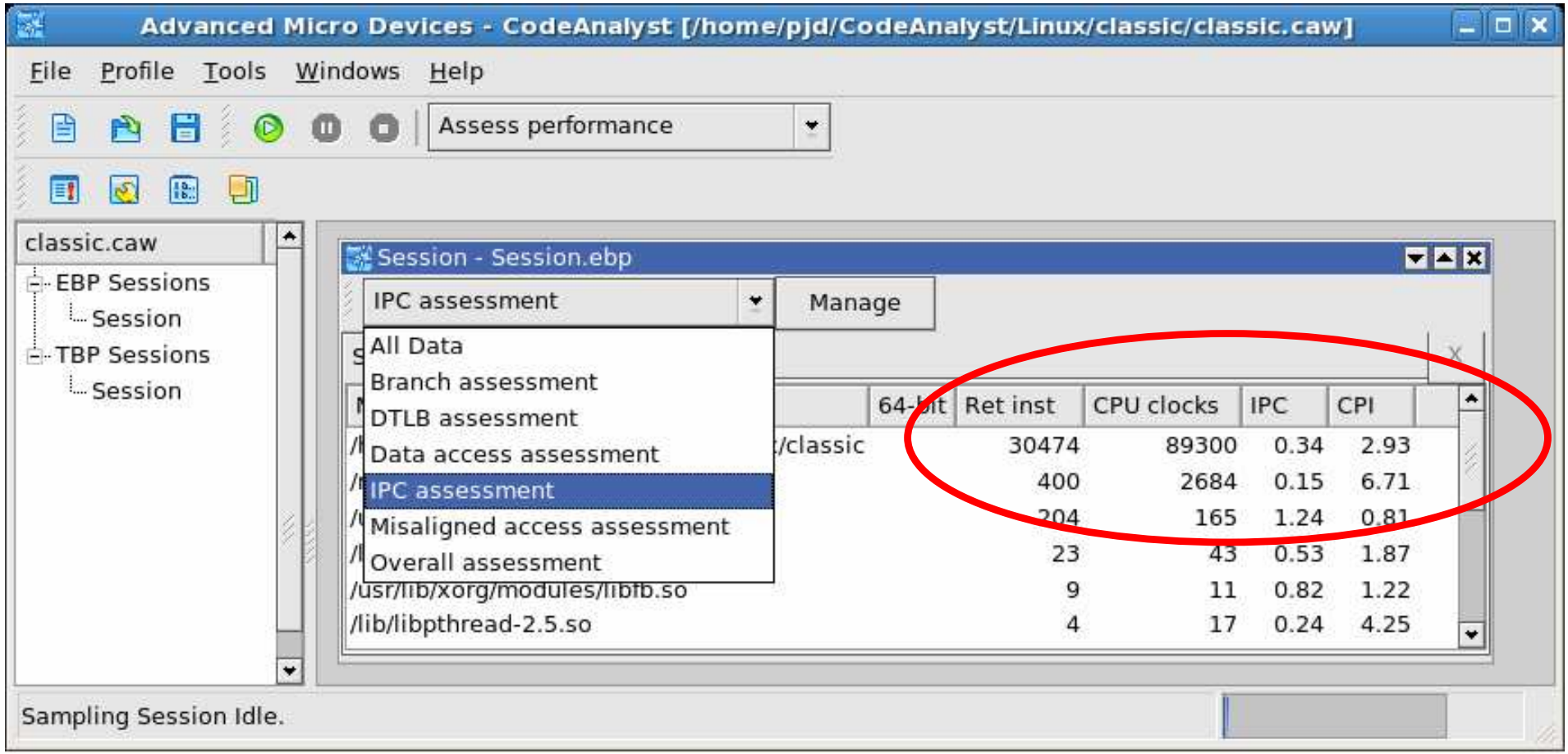
AMD Smarter Choice

- Choose from predefined profile configurations
- Define a customized configuration ("Current")

Profile configuration	Focus
Time-based profile	Hot-spot identification
Assess performance	Overall evaluation ("triage")
Investigate data access	Memory access patterns / locality
Investigate L2 cache access	Memory access patterns / locality
Investigate branching	Branch and return prediction
Investigate instruction access	Code placement / locality

Event-based profiling

- Views are offered depending upon available event data
- Views show computed measurements such as ratios
- IPC view shows “Instructions per cycle”



Event-based profiling

- DTLB view shows the DTLB miss rate and ratio
- DTLB miss rate is high – 1 miss per 10 instructions
- Low IPC is due to DTLB misses

Session - Session.ebp

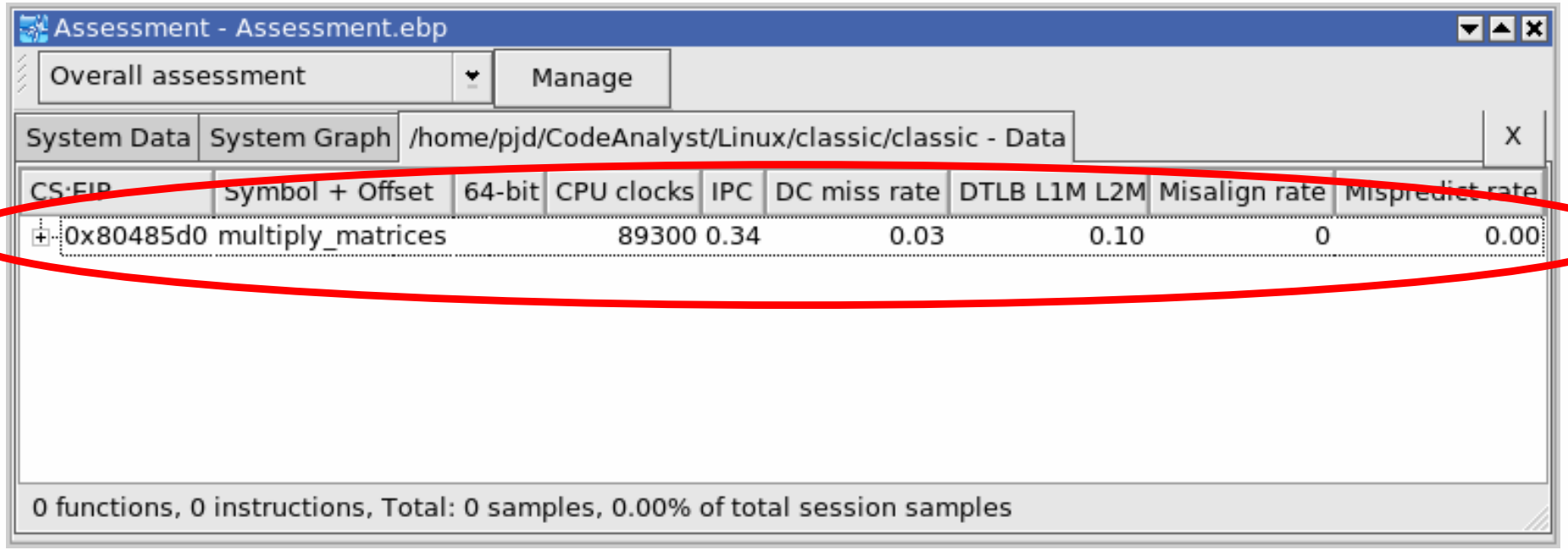
DTLB assessment | Manage

System Data | System Graph

Module Name	Ret inst	DC accesses	DTLB L1M L2M	DTLB request rate	DTLB L1M L2M rate	DTLB L1M L2M
/home/pjd/CodeAnalyst/Linux/classic/classic	30474	7784	30259	0.26	0.10	0.39
/no-vmlinux	499	147	19	0.37	0.00	0.01
/usr/local/bin/oprofiled	204	40	0	0.20	0	0
/lib/libc-2.5.so	23	7	1	0.30	0.00	0.01
/usr/lib/xorg/modules/libfb.so	9	2	0	0.22	0	0
/usr/lib/qt-3.3/lib/libqt-mt.so.3.3.7	9	4	0	0.44	0	0
/lib/libglib-2.0.so.0.1200.9	8	8	0	1	0	0
/usr/bin/Xorg	5	6	0	1.20	0	0
/usr/lib/xorg/modules/libshadow.so	4	2	0	0.50	0	0
/usr/lib/libX11.so.6.2.0	4	3	0	0.75	0	0
/lib/libpthread-2.5.so	4	2	0	0.50	0	0

Event-based profiling

- Drill down to functions within a module
- Find the function responsible for high DTLB miss rate



Event-based profiling

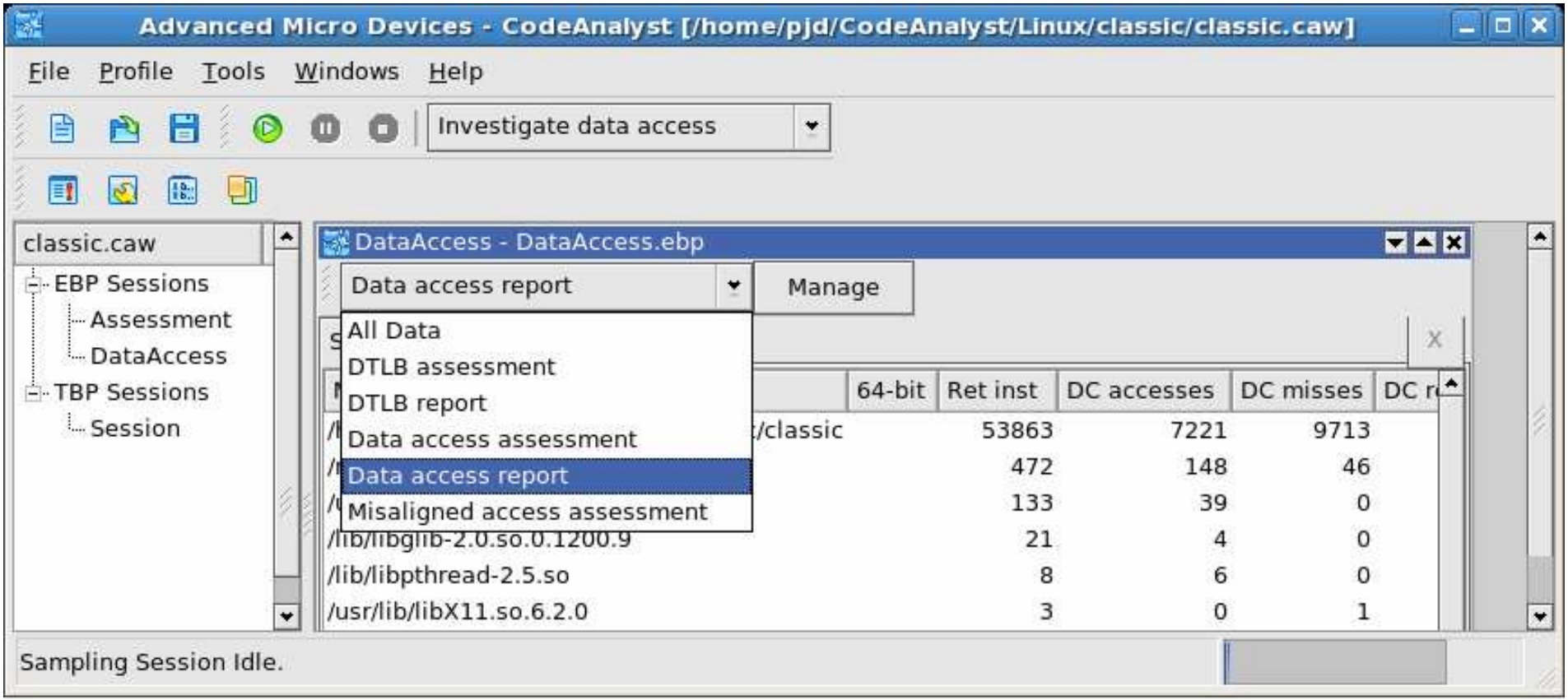
- Drill down to the culprit code region

Line	Source	Code	CPU clocks	IPC	DC miss rate	DTLB L1M L2M	Misalign
58	void multiply_matrices()		0	0	0	0	
59	{						
60	// Multiply the two matrices						
61	for (int i = 0 ; i < ROWS ; i++) {						
62	for (int j = 0 ; j < COLUMNS ; j++) {		3	1	0	0.07	
63	float sum = 0.0 ;						
64	for (int k = 0 ; k < COLUMNS ; k++) {		55628	0.33	0.03	0.10	
65	sum = sum + matrix_a[i][k] * m...		33074	0.35	0.03	0.09	
	flds (%eax)	d9 0	12304	0.40	0.03	0.09	
	fmuls (%ecx)	d8 09	10262	0.31	0.04	0.11	
	faddp %st,%st(1)	de c1	10508	0.34	0.03	0.09	
66	}						
67	matrix_r[i][j] = sum ;		584	0.33	0.03	0.09	
68	}						

0 Instruction, 0 samples, 0.00% of total

Event-based profiling

- Use focused profile configurations to investigate specific issues in more depth
- "Data access" concentrates on DC/DTLB behavior



Event-based profiling

- Example: Data access report
- Shows data cache miss rate and miss ratio

DataAccess - DataAccess.ebp

Data access report | Manage

System Data | System Graph

Module Name	Ret inst	DC access	DC miss	DC refills	DC access rate	DC miss rate	DC miss ratio	DC refill rate
/home/pjd/CodeAnalyst/Linux/classic/classic	53863	7221	9713	9712	0.27	0.04	0.13	0.04
/no-vmlinux	472	148	46	45	0.63	0.02	0.03	0.02
/usr/local/bin/oprofiled	133	39	0	0	0.59	0	0	0
/lib/libglib-2.0.so.0.1200.9	21	4	0	0	0.38	0	0	0
/lib/libc-2.5.so	21	5	0	0	0.48	0	0	0
/usr/lib/xorg/modules/libfb.so	16	5	0	0	0.62	0	0	0
/usr/bin/Xorg	11	6	0	0	1.09	0	0	0
/usr/lib/qt-3.3/lib/libqt-mt.so.3.3.7	10	2	2	0	0.40	0.04	0.10	0
/usr/lib/xorg/modules/libshadow.so	8	8	0	0	2	0	0	0

AMD
Smarter Choice

Tuning

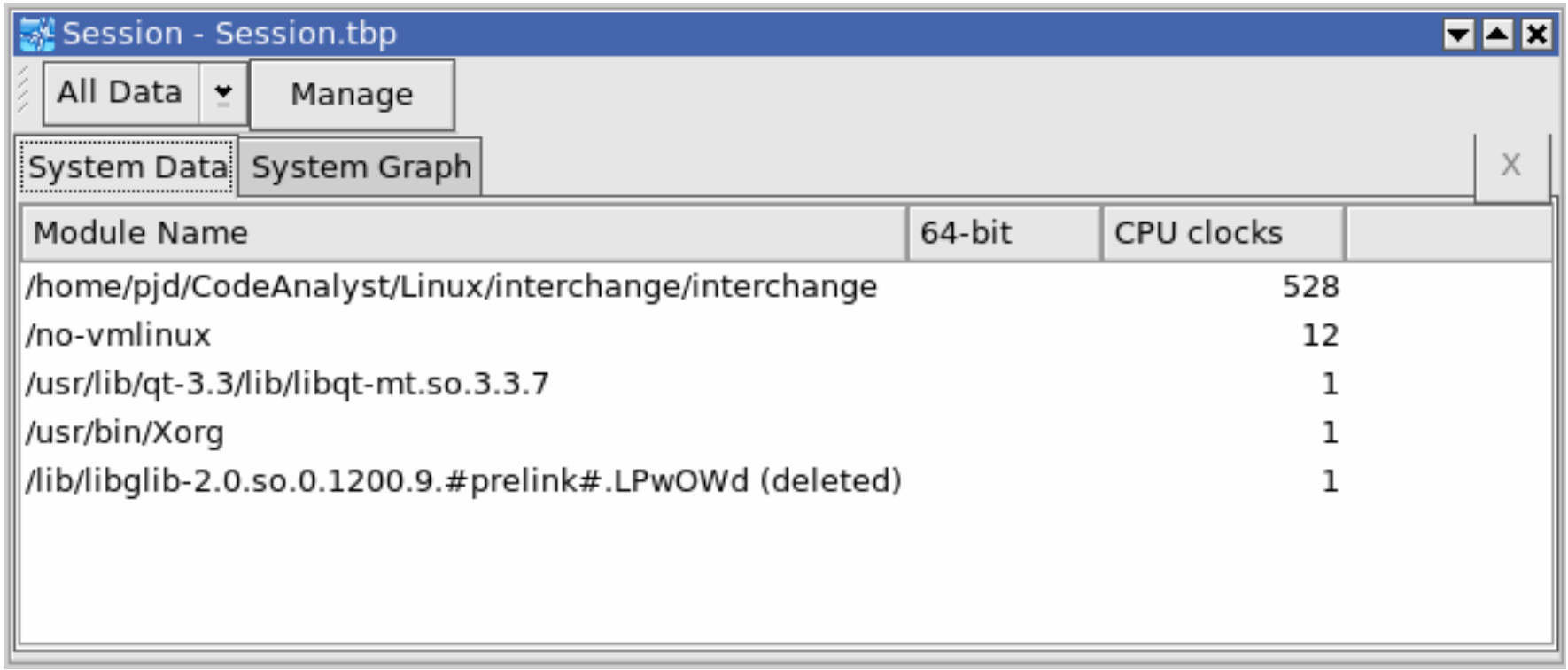
Make changes to tune the program

- Change the loop nest
- Use sequential unit strides to avoid DTLB/DC misses

```
void multiply_matrices()
{
  // Multiply the two matrices
  for (int i = 0 ; i < N ; i++) {
    for (int k = 0 ; k < K ; k++) {
      for (int j = 0 ; j < M ; j++) {
        matrix_r[i][j] = matrix_r[i][j] +
                 matrix_a[i][k] * matrix_b[k][j] ;
      }
    }
  }
}
```

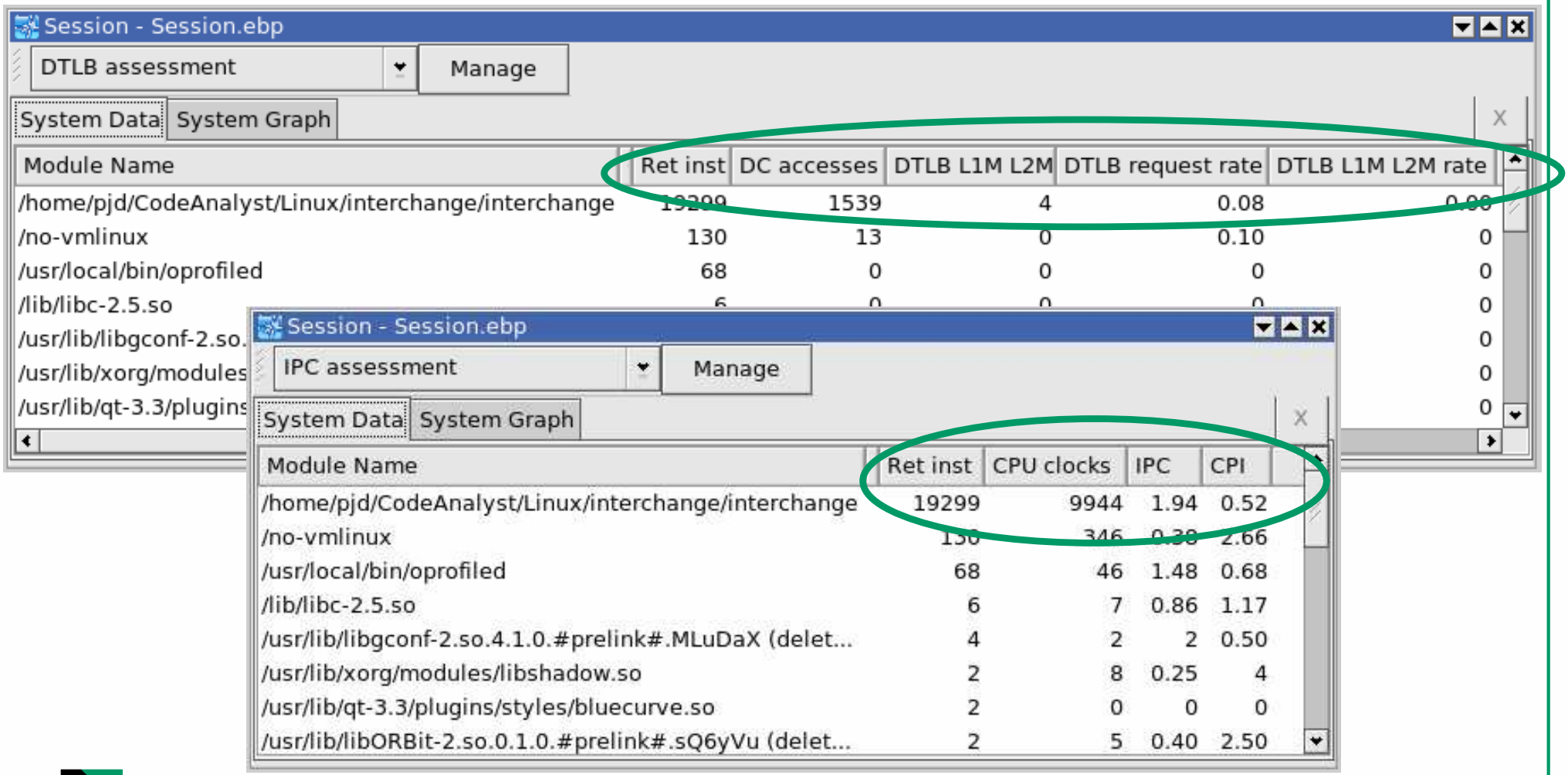
Measure again and evaluate results

- Execution time is reduced: 10.97 sec → 2.07 sec
- TBP samples are reduced: 8641 → 528



IPC is improved

- DTLB misses are reduced (4 vs. 30259) ...
- ... and IPC is improved (1.94 vs. 0.34)



How to get AMD CodeAnalyst

- Free!
- Download from AMD Developer Central
  - http://developer.amd.com/
  - http://www.amd.com/codeanalyst/
- Available versions
  - V2.5 tarball and RPMs: RHEL 4 (U1, U2, U3)
  - V2.6 tarball and RPMs: RHEL 5 (beta), Fedora Core 5 and 6
- E-mail: support.codeanalyst@amd.com

Resources

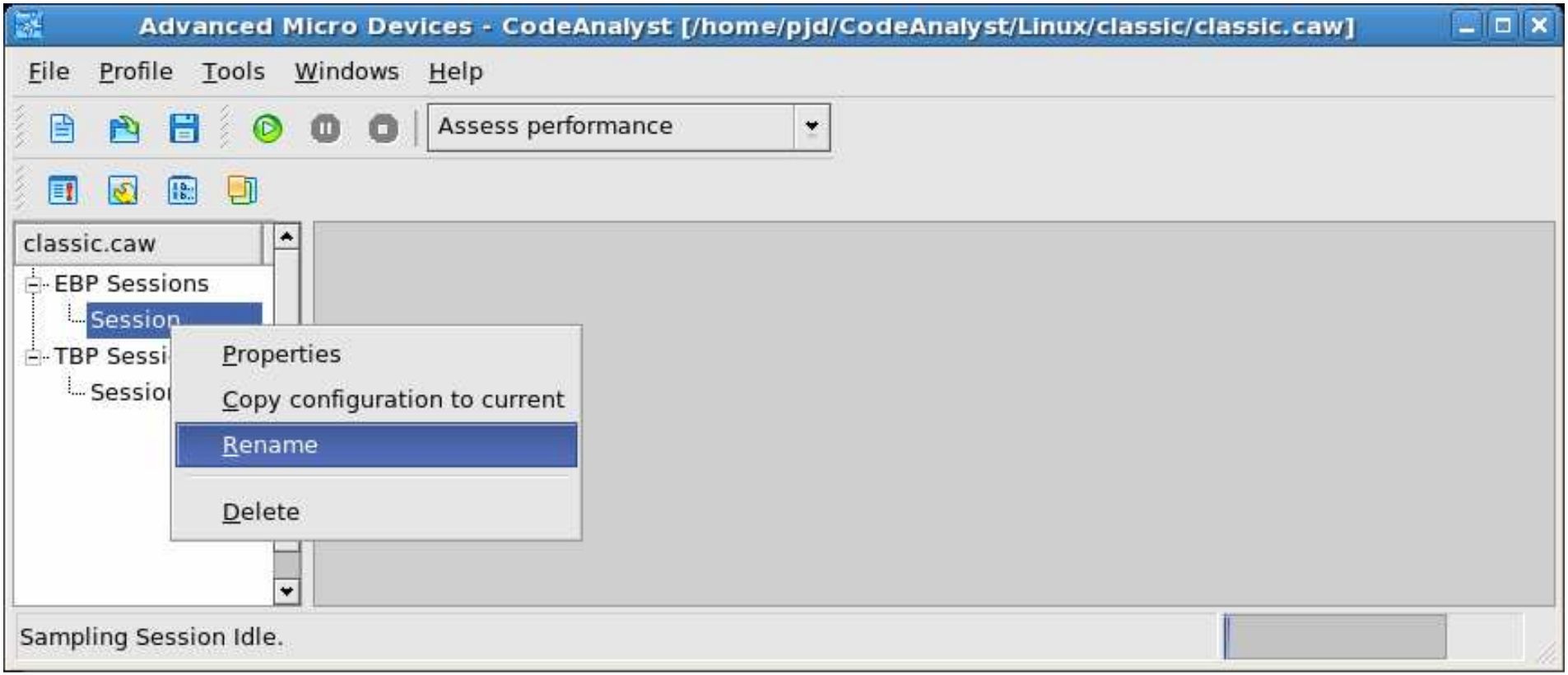
- Available at AMD Developer Central
- "An introduction to analysis and optimization with AMD CodeAnalyst"
  - http://developer.amd.com/articles.jsp?id=2&num=1
- "Basic performance measurements for AMD Athlon™ 64 and AMD Opteron™ Processors"
  - http://developer.amd.com/articles.jsp?id=90&num=1
- "BIOS and Kernel Developer's Guide" or "BKDG" for descriptions of hardware performance events
  - http://developer.amd.com/documentation.jsp
- Other AMD tools including performance libraries
  - http://developer.amd.com/downloads.jsp

AMD
Smarter Choice

Back-up slides

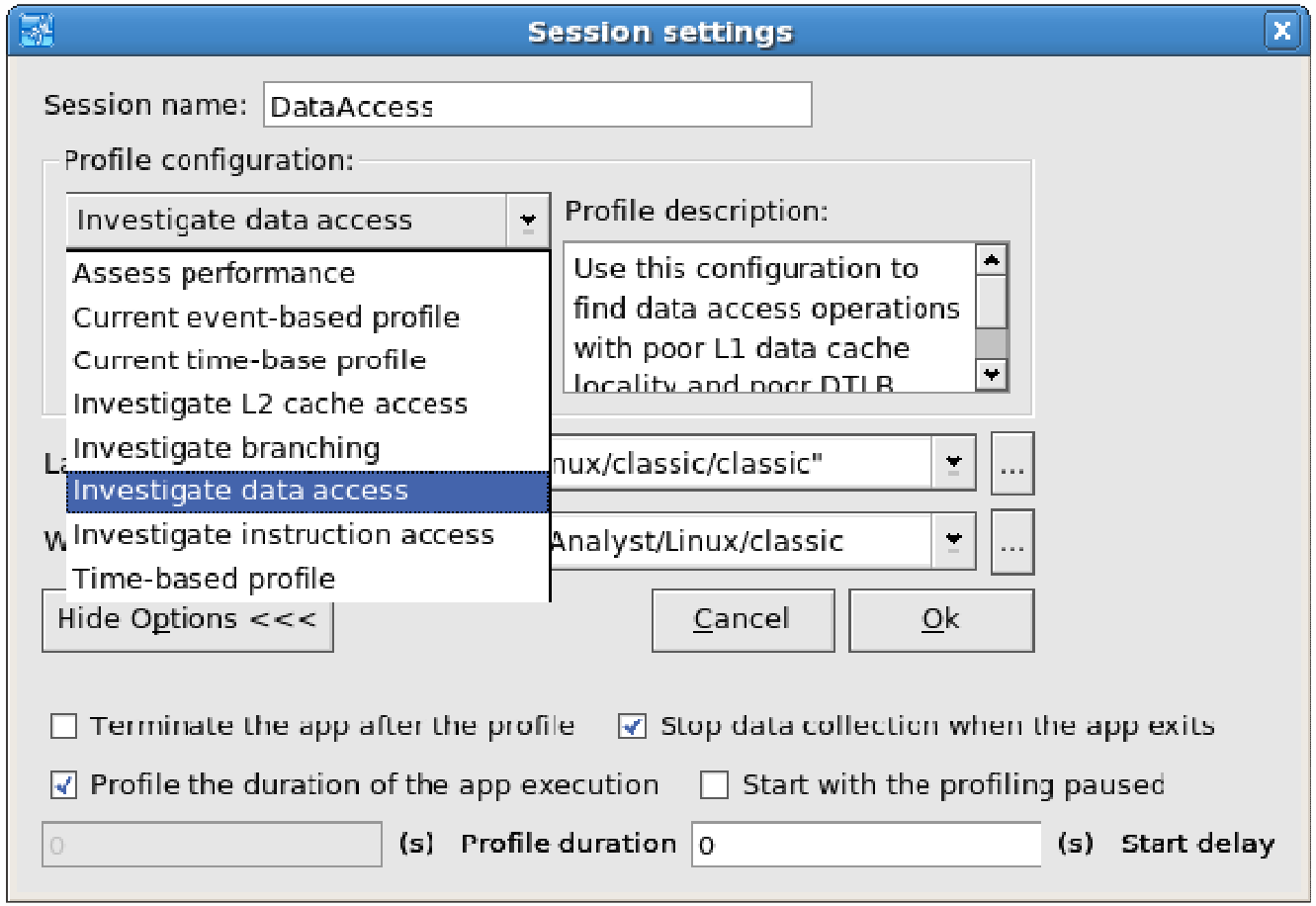
Rename / delete session data

- Right-click on session to name or delete it
- Can also show session properties (such as events)

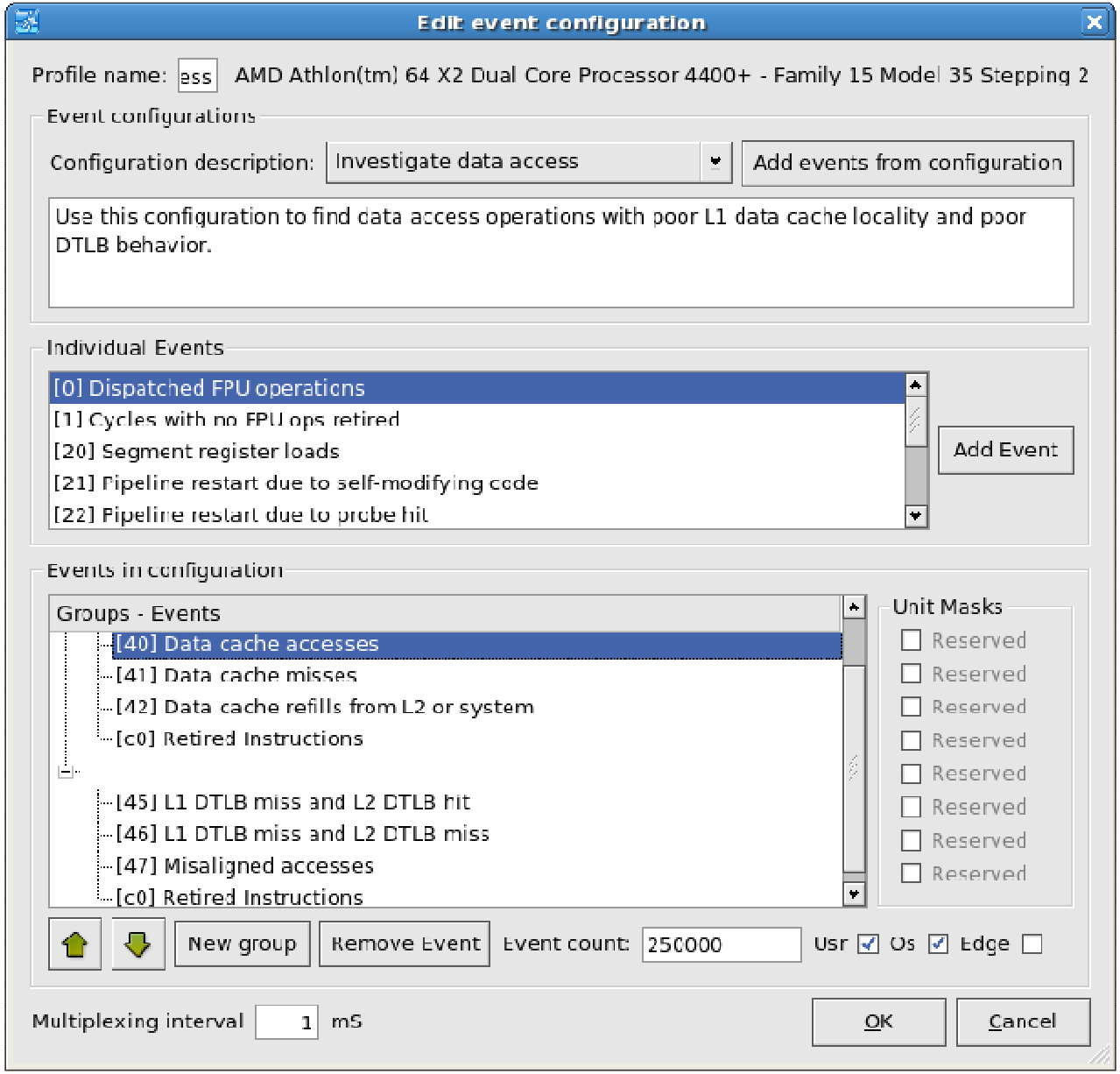


Change session settings

- Change session name
- Choose a profile (data collection) configuration
- Edit “current” profile configuration to customize

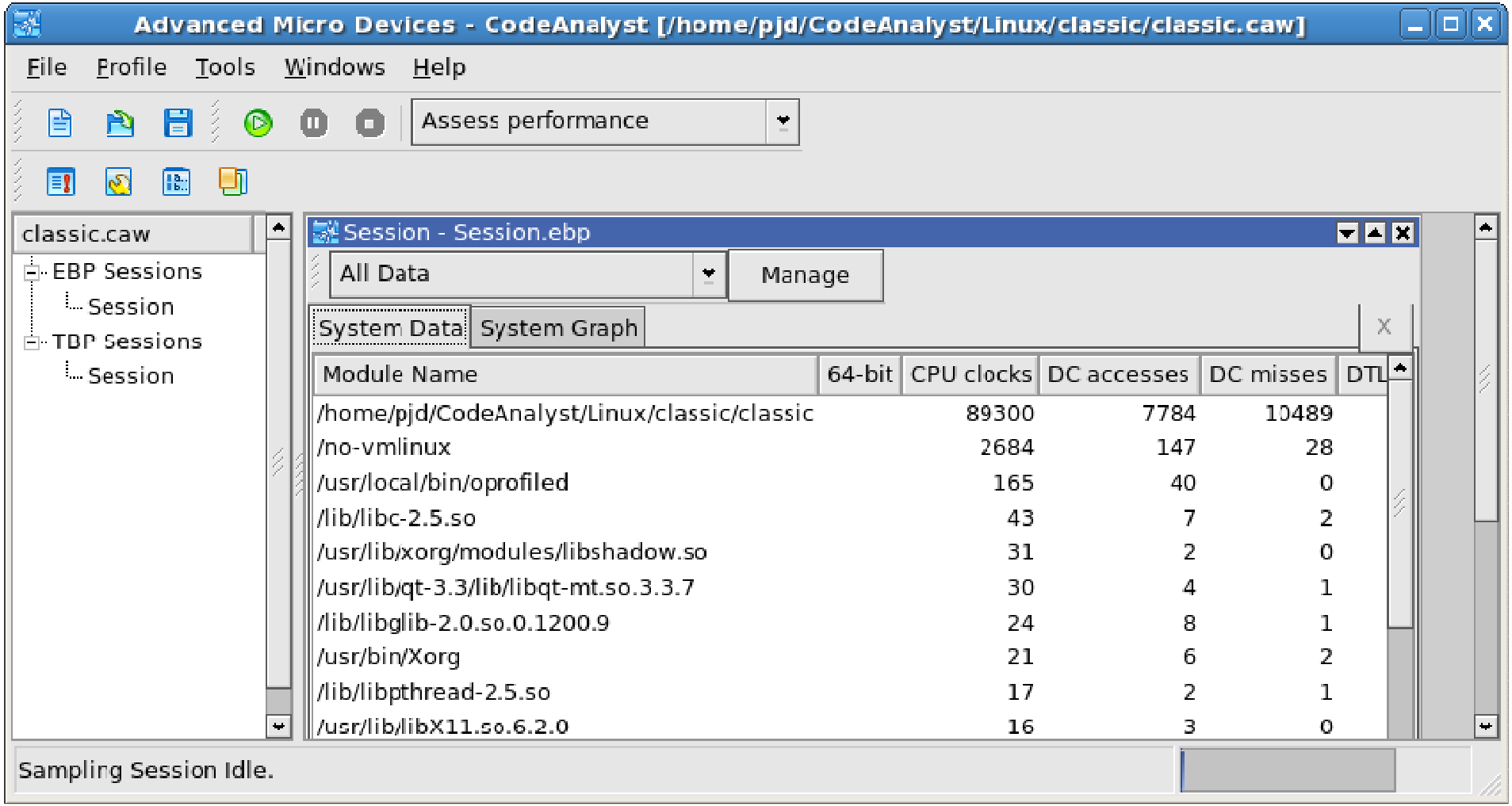


Edit profile configuration



Event-based profiling

- The “All data” view shows all event (sample) counts



Trademark Attribution